WEST

Generate Collection | Print

L5: Entry 4 of 7 File: USPT Oct 17, 2000

US-PAT-NO: 6134629
DOCUMENT-IDENTIFIER: US 6134629 A

TITLE: Determining thresholds and wrap-around conditions in a first-in-first-out memory supporting a variety of read and write transaction sizes

DATE-ISSUED: October 17, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
L'Ecuyer; Brian Peter	Elk Grove	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Hewlett-Packard Company	Palo Alto	CA			02

APPL-NO: 09/ 015415 [PALM]
DATE FILED: January 29, 1998

INT-CL: [07] G06 F 12/02

US-CL-ISSUED: 711/110; 711/219, 710/50, 365/221
US-CL-CURRENT: 711/110; 365/221, 710/50, 711/219

FIELD-OF-SEARCH: 711/110, 711/217, 711/218, 711/219, 710/52, 365/221

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4839866	June 1989	Ward et al.	365/221
☐	4891788	January 1990	Kreifels	365/49
☐	5668767	September 1997	Barringer	365/221

ART-UNIT: 279

PRIMARY-EXAMINER: Nguyen; Hiep T.

ABSTRACT:

Data is read from a first-in-first-out (FIFO) queue. A first condition flag is generated which indicates whether a read transaction of a first transaction size may be performed. When a write address for the FIFO queue is greater than a read address for the FIFO queue, the first condition flag is set to true when the read address plus

the first transaction size is less than or equal to the write address. When the write address for the FIFO queue is less than the read address for the FIFO queue, the first condition flag is set to true when the read address plus the first transaction size is less than the write address plus a maximum depth of the FIFO queue. A first read transaction of the first transaction size from the FIFO queue is performed only when the first condition flag is true.

11 Claims, 7 Drawing figures

WEST

☐ Generate Collection | Print

L5: Entry 6 of 7 File: USPT Mar 28, 2000

US-PAT-NO: 6044434
DOCUMENT-IDENTIFIER: US 6044434 A
**** See image for Certificate of Correction ****

TITLE: Circular buffer for processing audio samples

DATE-ISSUED: March 28, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Oliver; Richard J.	Laguna Beach	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Sony Corporation				JP	03
Sony Electronics, Inc.	Park Ridge	NJ			02

APPL-NO: 08/ 937140 [PALM]
DATE FILED: September 24, 1997

INT-CL: [07] G06 F 12/00

US-CL-ISSUED: 711/110; 711/219, 700/94
US-CL-CURRENT: 711/110; 700/94, 711/219

FIELD-OF-SEARCH: 711/110, 711/219, 700/94

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected | Search ALL

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5450544	September 1995	Dixon et al.	711/110
☐	5606707	February 1997	Tomassi et al.	700/259

ART-UNIT: 271

PRIMARY-EXAMINER: Lane; Jack A.

ATTY-AGENT-FIRM: Fulwider Patton Lee & Utecht, LLP

ABSTRACT:

A circular buffer in a system for processing audio samples wherein the buffer includes a sample window, the length of which is the length of a plurality of samples, the

length of the circular buffer is a multiple of the length of the sample window, and the entire sample window is treated as a contiguous linear address space on each iteration of the processing system, that is moved through the physical multiple sample length buffer between iterations of the processing system, and is reset to the beginning every sample window number of iterations of the processing system. The circular buffer substantially reduces the number of address calculations in processing systems where every buffer position is addressed on every iteration and where circular addressing is not provided in hardware.

10 Claims, 5 Drawing figures

WEST

☐ Generate Collection | Print

L9: Entry 1 of 30 File: PGPB Jun 5, 2003

PGPUB-DOCUMENT-NUMBER: 20030105917
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030105917 A1

TITLE: CIRCULAR ADDRESS REGISTER

PUBLICATION-DATE: June 5, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
OSTLER, FARRELL L.	ALBUQUERQUE	NM	US	
DAGHER, ANTOINE FARID	ALBUQUERQUE	NM	US	

APPL-NO: 09/ 466404 [PALM]
DATE FILED: December 17, 1999

CONTINUED PROSECUTION APPLICATION: This is a publication of a continued prosecution application (CPA) filed under 37 CFR 1.53(d).

INT-CL: [07] G06 F 12/06

US-CL-PUBLISHED: 711/110; 711/154, 711/220
US-CL-CURRENT: 711/110; 711/154, 711/220

REPRESENTATIVE-FIGURES: 1

ABSTRACT:

A device and corresponding programming instructions are provided that facilitate a circular addressing process. The device is configured to provide an address output that is constrained to lie within specified bounds. When a "circular increment" or "circular decrement" instruction is executed that would cause the address to exceed a bound, the address is reset to the other bound. In a preferred embodiment, the programming instruction also sets condition flags that indicate when the address is at each bound. By providing these "bounds" flags in conjunction with the circular addressing operation, multiple-word data items can be processed efficiently. A base-address of N contiguous words in a memory is loaded into the circular register, and a circular addressing instruction is used to access each word of the N contiguous words in sequence; a bounds flag is set when the last word of the multi-word data item is accessed. This bounds flag may thus be used to signal the end of processing of N words of a multi-word data item, and can be used to trigger a load of a next multi-word data item, or to trigger a next operation on the current data-item, and so on. Other condition flags are also provided to facilitate the processing of multi-word data-items.

WEST

Generate Collection Print

L9: Entry 1 of 30 File: PGPB Jun 5, 2003

DOCUMENT-IDENTIFIER: US 20030105917 A1
TITLE: CIRCULAR ADDRESS REGISTER

Abstract Paragraph (1):
A device and corresponding programming instructions are provided that facilitate a circular addressing process. The device is configured to provide an address output that is constrained to lie within specified bounds. When a "circular increment" or "circular decrement" instruction is executed that would cause the address to exceed a bound, the address is reset to the other bound. In a preferred embodiment, the programming instruction also sets condition flags that indicate when the address is at each bound. By providing these "bounds" flags in conjunction with the circular addressing operation, multiple-word data items can be processed efficiently. A base-address of N contiguous words in a memory is loaded into the circular register, and a circular addressing instruction is used to access each word of the N contiguous words in sequence; a bounds flag is set when the last word of the multi-word data item is accessed. This bounds flag may thus be used to signal the end of processing of N words of a multi-word data item, and can be used to trigger a load of a next multi-word data item, or to trigger a next operation on the current data-item, and so on. Other condition flags are also provided to facilitate the processing of multi-word data-items.

Current US Classification, US Primary Class/Subclass (1):
711/110

Summary of Invention Paragraph (5):
[0004] Circular lists are often used to implement First-In, First-Out (FIFO) buffers. The FIFO model represents a list of items wherein new items are added to the top of the list, and old items are removed from the bottom of the list. In effect, the FIFO is structured as a list of L data items; when a data item is added to the list, it is given the next available address of L+1; when a data item is removed from the list, the address of each data item remaining in the list is decremented by one. The moving of each remaining data item down in the list each time a data item is removed from the list, however, is inefficient. Instead, the defined "bottom" of the list is moved up by one each time a data item is removed. The logical address of each data-item in the list is defined by its distance from this defined bottom of the list, rather than by its physical location in the continually growing list. Even though the "removed" data items may remain in the physical locations below the "bottom" of the list, they are no longer within the logical address space of the list, and are no longer considered to be a part of the list of L items. As would be evident to one of ordinary skill in the art, the continued advancement of the physical location used for each added data item, without a corresponding physical relocation of each data item to lower physical locations as data items are removed, will eventually exhaust the physical resources used to contain these data items.

Summary of Invention Paragraph (7):
[0006] Typically, multiple programming instructions are required to control the incrementing and decrementing of a pointer that is associated with a circular list. The instructions must test whether the pointer has gone beyond the upper or lower bounds of the physically contiguous memory locations, and adjust the pointer accordingly to provide the circular addressing effect.

Summary of Invention Paragraph (10):
[0008] These objects and others are achieved by providing a device and corresponding

programming instructions that facilitate a circular addressing process. The device is configured to provide an address output that is constrained to lie within specified bounds. When a "circular increment" or "circular decrement" instruction is executed that would cause the address to exceed a bound, the address is reset to the other bound. In a preferred embodiment, the programming instruction also sets condition flags that indicate when the address is at each bound. By providing these "bounds" flags in conjunction with the circular addressing operation, multiple-word data items can be processed efficiently. A base-address of N contiguous words in a memory is loaded into the circular register, and a circular addressing instruction is used to access each word of the N contiguous words in sequence; a bounds flag is set when the last word of the multi-word data item is accessed. This bounds flag may thus be used to signal the end of processing of N words of a multi-word data item, and can be used to trigger a load of a next multi-word data item, or to trigger a next operation on the current data-item, and so on. Other condition flags are also provided to facilitate the processing of multi-word data-items.

Detail Description Paragraph (3):
[0013] FIG. 1 illustrates a processing system 100 that includes a circular-address-register 200 that is used to access a memory 110 comprising contiguous storage elements d.sub.0-d.sub.N-1 111-119 via a circular addressing scheme. The circular-address-register 200 provides an address 101 to the contiguous storage elements 111-119, corresponding, for example, to the aforementioned circular list. The address 101 contained in the circular-address-register 200 can be incremented 102 or decremented 103 to advance the pointer up or down the elements 111-119, respectively. The circular-address-register 200 is configured to determine an associated upper and lower bound corresponding to the range of the addresses of the memory elements 111-119. In a preferred embodiment, a base address A.sub.d 121 and a size N 122 are provided when the address register 200 is configured for processing the memory elements 111-119 as a circular list or buffer.

Detail Description Paragraph (5):
[0015] Note that the numeral "1" is used herein for ease of illustration. As is common in the art, the size of a "word" may differ from the scale that is used for addressing the elements d.sub.0-d.sub.N 111-119. That is, for example, the address 101 may be specified in terms of 8-bit bytes or 16-bit words, while the elements d.sub.0-d.sub.N-1 111-119 may be configured to contain 32-bit words. As used herein, the term "word" is defined to be the size of the elements d.sub.0-d.sub.N-1 111-119, and the increment and decrement of the address 101 contained in the circular-address-register 200 corresponds to a change of the address 101 by one word size.

Detail Description Paragraph (6):
[0016] A variety of embodiments of a circular-address-register that effects the above operations will be evident to one of ordinary skill in the art in view of this disclosure. In a preferred embodiment, both the circular increment operation and the circular decrement operation are provided, although a minimal embodiment may include one or the other. Also in a preferred embodiment, other operations are provided, as discussed below, to facilitate alternative and/or conventional uses of the circular-address-register 200. Note that a direct embodiment of the circular increment and decrement operations as defined above requires that the base address A.sub.d 121 and N 122 be stored, for comparison with the current address 101 contained in the circular-address-register 200.

Detail Description Paragraph (12):
[0022] Operations (3) "Circular Increment" implements a circular increment addressing function. When the index is at its upper bound (index=N-1), the next circular increment operation (3) resets the address Aout 231 to the lower address bound, by subtracting N-1 from its current value, and resets the index to zero, corresponding to the word at the lower address bound. Otherwise, when the index is not at its upper bound, the next circular increment operation (3) increments the current contents of the address register and index register.

Detail Description Paragraph (13):
[0023] Operations (4) "Circular Decrement" implements a circular decrement addressing function. When the index is at its lower bound (index=0), the next circular decrement

operation (4) resets the address Aout 231 to the upper address bound, by adding N-1 to its current value, and resets the index to N-1, corresponding to the word at the upper address bound. Otherwise, when the index is not at its lower bound, the next circular decrement operation (4) decrements the current contents of the address register and index register.

Detail Description Paragraph (14):
[0024] Operations (5) "Increment" implements a modified form of a conventional increment operation. The increment operation (5) increments Aout 231, without regard to an upper address bound, similar to a conventional register-increment operation. The increment operation (5) modifies the index in the same manner as the circular increment operation (3), thereby performing a "modulo N" up-counting function via the index.

Detail Description Paragraph (15):
[0025] Operations (6) "Decrement" implements a modified form of a conventional decrement operation. The decrement operation (6) decrements Aout 231, without regard to a lower address bound, similar to a conventional register-decrement operation. The decrement operation (6) modifies the index in the same manner as the circular decrement operation (4), thereby performing a "modulo N" down-counting function via the index.

Detail Description Paragraph (21):
[0031] The foregoing merely illustrates the principles of the invention. It will thus be appreciated that those skilled in the art will be able to devise various arrangements which, although not explicitly described or shown herein, embody the principles of the invention and are thus within its spirit and scope. For example, as illustrated in FIG. 2, an optional Aout=? 0 flag 214 may be provided to facilitate an efficient counter function in conjunction with the "Load_L" and "Decrement" or "Increment" operations. That is, the content of the address register 230 need not be an address pointer, and may be loaded with a count value M. Subsequent decrement operations will result in the Aout=? 0 being set, signaling a completion of M counts. These and other system configuration and optimization features will be evident to one of ordinary skill in the art in view of this disclosure, and are included within the scope of the following claims.

Detail Description Table CWU (1):
1 OPERATION EFFECT Circular If Address >=? A.sub.d + Then Address = A.sub.d Increment (102) (N - 1) Else Address = Address + 1. Circular If Address <=? A.sub.d Then Address = A.sub.d +(N - 1) Decrement (103) Else Address = Address - 1.

Detail Description Table CWU (2):
2 (1) Load_L Aout = Ain; Index = 0. (2) Load_U Aout = Ain; Index = N - 1. (3) Circular If (Index =? (N - 1)) Then Aout = Aout - (N - 1); Increment Index = 0 Else Aout = Aout + 1; Index = Index + 1. (4) Circular If (Index =? 0) Then Aout = Aout + (N - 1); Decrement Index = (N - 1) Else Aout = Aout - 1; Index = Index - 1. (5) Increment Aout = Aout + 1; If (Index =? (N - 1)) Then Index = 0 Else Index = Index + 1. (6) Decrement Aout = Aout - 1; If (Index =? 0) Then Index = (N - 1) Else Index = 0.

CLAIMS:

1. A processing system comprising: a processor that is configured to execute program instructions, a circular-address-register having an associated upper and lower bound, that is configured to receive commands from the processor, the commands comprising at least one of: a circular increment command that is configured to increment a content of the circular-address-register and to reset the content to the lower bound after the content reaches the upper bound, and a circular decrement command that is configured to decrement the content of the circular-address-register and to reset the content to the upper bound after the content reaches the lower bound.

8. The processing system of claim 7, wherein the commands further include at least one of: an increment command that is configured to increment a content of the circular-address-register, independent of the upper bound, and a decrement command that is configured to decrement the content of the circular-address-register, independent of the lower bound.

14. The circular-address-register of claim 13, wherein the command processor is configured to update the address register, when the register-command is at least one of a circular-increment-command and a circular-decrement-command, by: adding a first amount to the address, and resetting the index to a predetermined value, when the index equals a bound value, and adding a second amount to the address and to the index, when the index does not equal the bound value.